7-31-2020

Implementing and Testing VLANs in Meshed Tree Protocol

Ankush Kaul
ak6624@g.rit.edu

Implementing and Testing VLANs in Meshed Tree Protocol

by Ankush Kaul

Committee Members

Prof. Nirmala Shenoy

Prof. Bruce Hartpence

Prof. Bill Stackpole

Prof. Daryl Johnson

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Networking and Systems Administration

Department of Information Sciences and Technologies

B. Thomas Golisano College of Computing and Information Sciences

Rochester Institute of Technology

July 31, 2020

Abstract

Meshed Tree Protocol (MTP)[1][2] is being developed to overcome the performance challenges faced with older loop avoidance layer 2 protocols like Spanning Tree Protocol (STP)[3] and Rapid STP (RSTP)[5] which have high convergence time, causing a significant delay during initial convergence and subsequent re-convergence in the event of topology change or link failure. This slow convergence is not suitable for the modern high speed and dynamic networks and is being addressed with better performing protocols like MTP which uses Meshed Tree Algorithm (MTA) to form multiple overlapping trees[1]. In this thesis we will implement Virtual Local Area Network(VLAN)[4] for MTP networks using the Global Environment for Network Innovation (GENI)[19] testbed.

With the growing size and complexity of modern day networks it is essential to segment a larger network into isolated smaller sections which improve the security, reliability, and efficiency of the network. This is achieved by using VLANs which act as separate smaller networks within a larger network. In this thesis we will discuss the working and benefits of VLANs in the current implementations of STP and how can VLANs be introduced in MTP along with a basic implementation of VLANs in the code developed for MTP by extending it to support Multi Meshed trees, where each meshed tree would cover a VLAN.

Contents

List of Figures

List of Tables

1 Introduction

A layer 2 network has to be highly reliable in forwarding data frames and fast in handling any link or node failures. This is achieved by having a meshed network topology with multiple redundant paths for frames to flow so that in case of any link or path failures the data continues to reach its destination via different paths. This method has a huge drawback of loop formation which can lead to broadcast storms [23] and cause the network to degrade or stop working completely. To address this issue loop avoidance protocols like Spanning Tree Protocol (STP) and Rapid Spanning Tree Protocol (RSTP)[10][17] are used which create loop free forwarding paths in such a way that each node is included but there is only one path to each node. This is achieved by finding loops in the network and then blocking a port on this loop based on predetermined conditions and priorities. The loop detection process introduces some problems as each time there are any topology changes or link failures, there is a need to find the network loops again and populate loop free forwarding trees. After link failure detection, the protocols take a significant amount of time to repair the tree and then start forwarding frames.

STP protocol used to take around 40-50 seconds for re-convergence after failure before it could stabilize and start forwarding frames which was unacceptable in modern networks which requires high uptime. RSTP, which replaced STP, is the most popular loop avoidance protocol being used currently in customer switched networks. RSTP tries to address this delay in convergence issue by reducing the time intervals and faster tree calculation but as the networks and applications continue to evolve we continue to look for near real time loop avoidance methods. This delay in convergence can cause significant packet drops and delays which in modern high speed and high available infrastructures can cause downtime impacting the business. The root election process in STP/RSTP also adds a significant initial delay for the protocol convergence as the root election process requires all switches to share their Bridge ID, which is a combination of Bridge Priority value and MAC address of the switch, with each other to decide which switch will be elected as root. This automatic root election process is not efficient as it does not elect the best performing or centrally located switch as root which would be an ideal selection, hence in all well designed networks the root is manually selected by setting the Bridge Priority.

Meshed Tree Protocol (MTP) which is based on Meshed Tree Algorithms (MTA) is designed to completely eliminate the initial root election time as the root node is set manually thus eliminating the automatic root election process. MTA also allows very fast re-convergence after a link failure by using multiple trees from a single root. As MTP node has multiple paths already configured, in case of failure it can use a backup path to forward frames almost instantly. Details or working of this protocol is described in upcoming sections of this paper. As we have seen in various papers published the MTP provides a significant improvement over currently popular RSTP. One of the main drawbacks of MTP is root redundancy which was addressed with the implementation of Multi Meshes Tree Protocol (MMTP). The current MMTP implementation introduces root redundancy by allowing MTP to run with two roots and in case one root fails the other root can take over. Also as current switched networks use VLANs to isolate user traffic, MTP implementations so far do not have this capability. In this thesis we propose a technique to support Virtual Local Area Network (VLAN)[4] in switched networks running MTP which will allows us to segregate network traffic.

Introduction of VLAN in MTP can provide multiple benefits like security, network management, efficiency and improved scalability. This is achieved by allowing users to configure VLAN on switch interfaces and then restricting MTP traffic of that interface to the VLAN hence creating separate virtual subnets within the network. Keeping the traffic separate will make the network more secure as users within a VLAN are able to broadcast to a restricted community. This will also have an added benefit of improved efficiency and scalability as you can limit the number of hosts and network devices within a VLAN.

1.1 Project Objective

The objective of this thesis was to discuss the benefits of VLANs in Meshed Tree Protocol which will help address issues like security, scalability and network management and implement a basic VLAN functionality for MTP. We used the current Multi Meshed Tree based MTP code as our starting point to implement a basic VLAN functionality and test it on multiple topologies. The expected output will help us show that we were able to successfully implement VLAN functionality and segregate the network broadcast traffic generated by MTP into separate VLANs.

We used Global Environment for Network Innovation (GENI) [19] as the testbed to demonstrate how VLANs in MTP can be used to restrict the broadcast traffic and tree formation to nodes within the VLAN. GENI provides us a flexible environment with capability to create a connected network consisting of multiple hosts. These hosts can be used to develop, deploy and run the MTP code for the VLAN implementation in MTP.

The first step is to introduce a single VLAN and add the functionality to read a VLAN configuration file provided by the user which has to be in a specific predetermined format. This VLAN information is stored in a table and it gives the user the flexibility to decide what nodes and interfaces will be a part of the VLAN. These pre-configured interfaces will be allowed to send broadcast traffic for this VLAN hence limiting the broadcast to other nodes in the network. This is then tested by introducing a single VLAN and running the MTP code to generate a Meshed Tree that include only the configured interfaces in that VLAN.

Next we expand this functionality to multiple VLANs, we achieve this by enabling creation of multiple VLAN tables using the config file and using these tables to send MTP messages in different VLANs. For this thesis we have implemented two VLANs along with the Multi Meshed Tree based MTP to separate our network into two subnets and then build the Meshed Trees for each VLAN.

2 Background

Loop Avoidance protocols are being continuously researched and modified for improving convergence times and performance and to adopt it to modern network requirements. Introduction of RSTP significantly improved convergence times which we try to further improve with MTP. Similarly, with increased use of a number of VLANs in current networks, Multiple Spanning Tree Protocol (MSTP) and Per VLAN Spanning Tree protocol (PVST) were introduced. Below we discuss VLANs and some of these loop avoidance protocols along with their implementation.

2.1 Virtual Area Network (VLAN)

All modern businesses small and big make use of local computer networks for various purposes like communication, accessing resources, developing products etc. As the local computer networks grow in size and complexity and are being used by multiple teams across an organization it becomes necessary to better secure and manage the network and the network resources. Using Virtual Local Area Network (VLAN) provides the ability to segment larger networks into smaller virtual subnets making them secure and easier to manage as they can be individually configured. VLAN allows different devices to be virtually grouped in the same network and share the same broadcast domain. This segmentation of the network helps improve security, reliability and efficiency of larger networks.

Security is one of the most important benefits of implementing VLAN as it prevents unauthorized access of network resources across different VLANs reducing risks and protecting user data. VLANs can be configured individually to have different security rules and restrictions. Apart from improved security, VLAN also provides better efficiency and network management by dividing a larger network into multiple smaller subnets makes it easier for the network administrator to set up, manage and troubleshoot individual VLANs. These smaller virtual networks also limit the amount of traffic flowing in the network improving the efficiency. These benefits led to VLAN being implemented widely in current computer networks thus introducing the need of modifying the loop avoidance protocols to be incorporated within each VLAN.

2.2 Loop Avoidance in Spanning Tree Protocols

Layer 2 loops continue to be a challenge whenever you want to introduce redundancy or load balancing in your network as the basic layer 2 switching cannot detect and prevent loops. Currently Rapid Spanning Tree Protocol (RSTP; IEEE 802.1w)[10] is the most widely used protocol for loop avoidance in Local Area Networks that was developed to address some major drawbacks of STP, which was the first loop avoidance protocol based on the Spanning Tree Algorithm (STA) proposed by Radia Perlman[8]. RSTP can be seen as an evolution of the 802.1D standard, which improves the convergence times for the spanning tree creation to a certain extent. STP uses Bridge Protocol Data Unit (BPDU) messages to share information between switches and detect network loops. Based on the information from the BPDU messages a root switch is elected (other switches are designated as non-root), which then helps identify various ports and block a selected port to break the loop. Below are explained some of the features and properties of STP.

Port States

Name	Description
Blocking	Port in the blocking state does not participate in frame forwarding and discards received frames
Listening	Port receives BPDUs from the network segment and directs them to the switch system module for processing
Learning	Port begins to process user frames and start updating the MAC address table
Forwarding	Port receives and send frames
Disabled	Port in the disabled state does not participate in frame forwarding or in STP operation

Table 2.1: Different Port States in STP

STP Port Roles

- **Root Port** is the port on a switch that is the closest way (Lowest Cost) to the Root Bridge.
- **Designated Port** is the port on a non-root switch that can send the best BPDU on its segment.
- **Non-Designated Port** is the port on a non-root switch that cannot send BPDU.

- **Disabled Port** is administratively down port that does not take part in STP

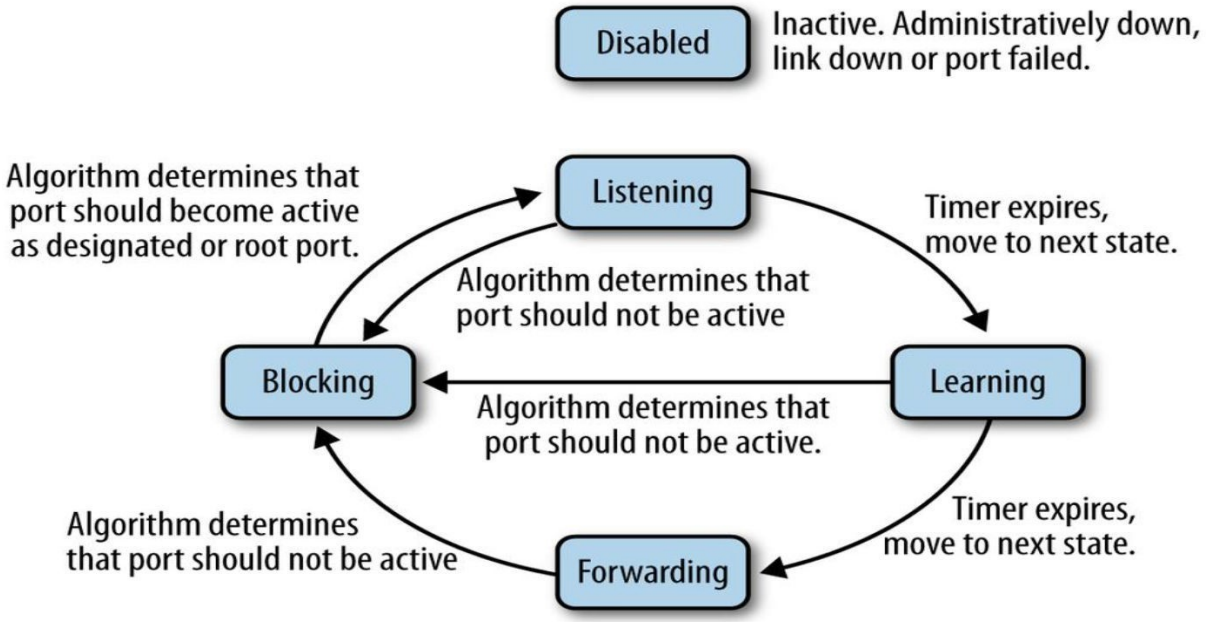


Figure 2.1: STP Port Transaction

BPDU Format

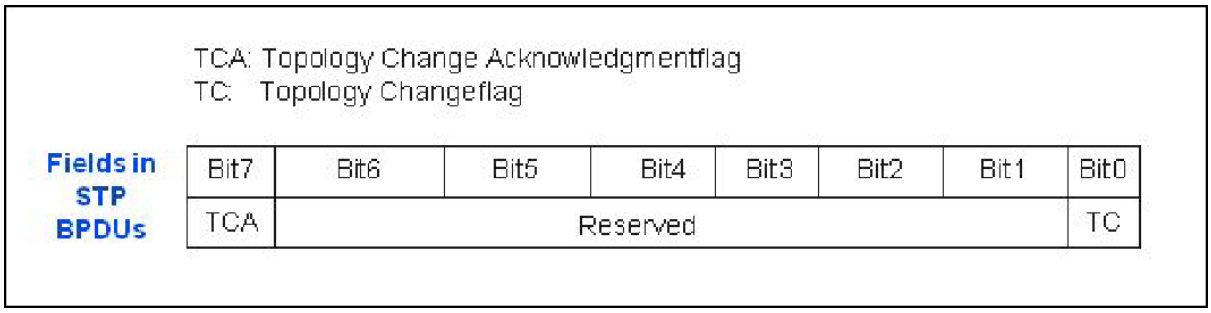


Figure 2.2: Fields in STP BPDU

Root Switch Election

Root Switch/Bridge is the central node for STP convergence [17]. The root switch election happens by comparing the Switch IDs which are shared with each other using BPDUs. The Switch ID is 8 byte value, the first 2 bytes is the Bridge Priority which can be configured on the switch, and the second part is the System ID which is the MAC address of the switch. Once BPDU messages are shared between all switches the one with the lowest Switch ID becomes the root switch.

STP Convergence and Port Blocking

All ports on the root bridge are in forwarding mode. On each active link a root port and a designated port are defined by STP. Root ports are ports on a switch which has the lowest path cost to the root bridge. A designated port is selected per segment and it's the port with the lowest Root Path Cost, BID, port ID. Any ports that are neither a root port nor a designated port are put into blocking mode thus breaking the switching loop.

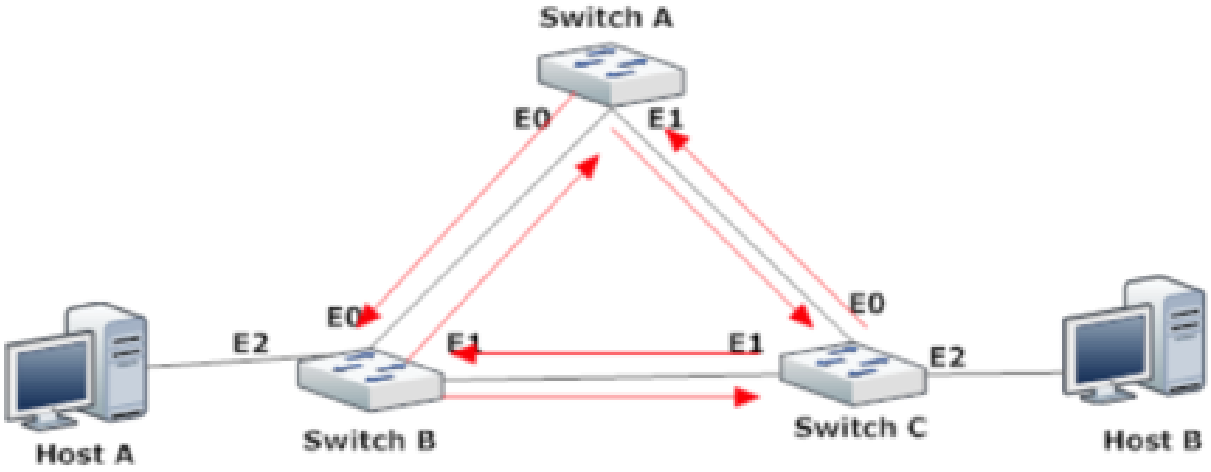


Figure 2.3: STP Convergence

Fig 2.3 shows the BPDU exchnage between switches and in Fig 2.4 after STP is fully converged the Port E1 on Switch C is put into blocking mode which breaks the loop.

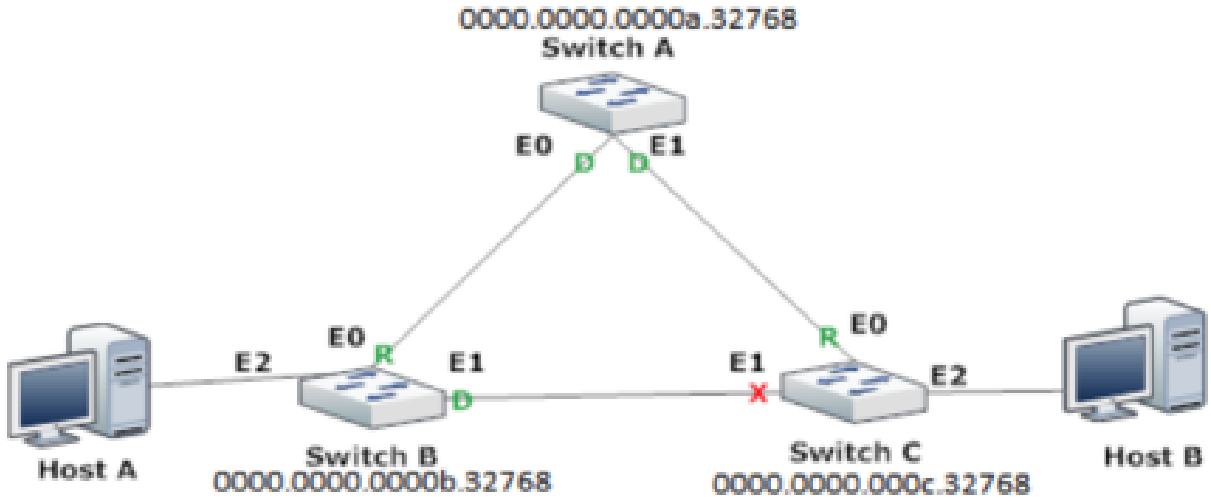


Figure 2.4: STP Convergence

The way STP is configured, it takes a long time in the range of 30-50 seconds for convergence and recalculation of the spanning tree if there is a topology change. Another major drawback of STP is that it is inefficient in bandwidth management as all the data

flows through one tree, the data takes only the path that is active and the redundant path is not utilized.

2.3 RSTP Upgrades

With advancement in network technologies the latency in convergence time for STP became unacceptable. This led to development of RSTP, which was then standardized by IEEE in 2001 as 802.1w[5]. RSTP tries to address the main drawback of STP by reducing the convergence time[10] by handling the topology changes is a more efficient way. RSTP achieves this by bypassing the Blocking State and Listening State of STP, thus achieving Forwarding State in under 10 seconds reducing the STP convergence time to less than half[17].

There are only three port states left in RSTP that correspond to the three possible operational states. The 802.1D disabled, blocking, and listening states are merged into a single 802.1w discarding state.

Port States

STP Port State	RSTP Port State	Port Included in Active Topology?	Port Learning MAC Addresses?
Blocking	Discarding	No	No
Listening	Discarding	No	No
Learning	Discarding	Yes	No
Forwarding	Learning	Yes	Yes
Disabled	Forwarding	Yes	Yes

Table 2.2: RSTP and STP Port States Comparison

RSTP Port Roles

- **Root Port** is the port on a switch that is the closest way (Lowest Cost) to the Root Bridge.
- **Designated Port** is the port that can send the best BPDU on its segment.
- **Alternate Port** is a blocking port that receives better BPDU from another switch. It is the backup of Root Port.
- **Backup Port** is a blocking port that receives better BPDU from the same switch. It is the backup of Designated Port.

- **Disabled Port** is administratively down port that does not take part in STP.
- **Edge Port** is port connected to host that does not receive a BPDU.

New BPDU Format

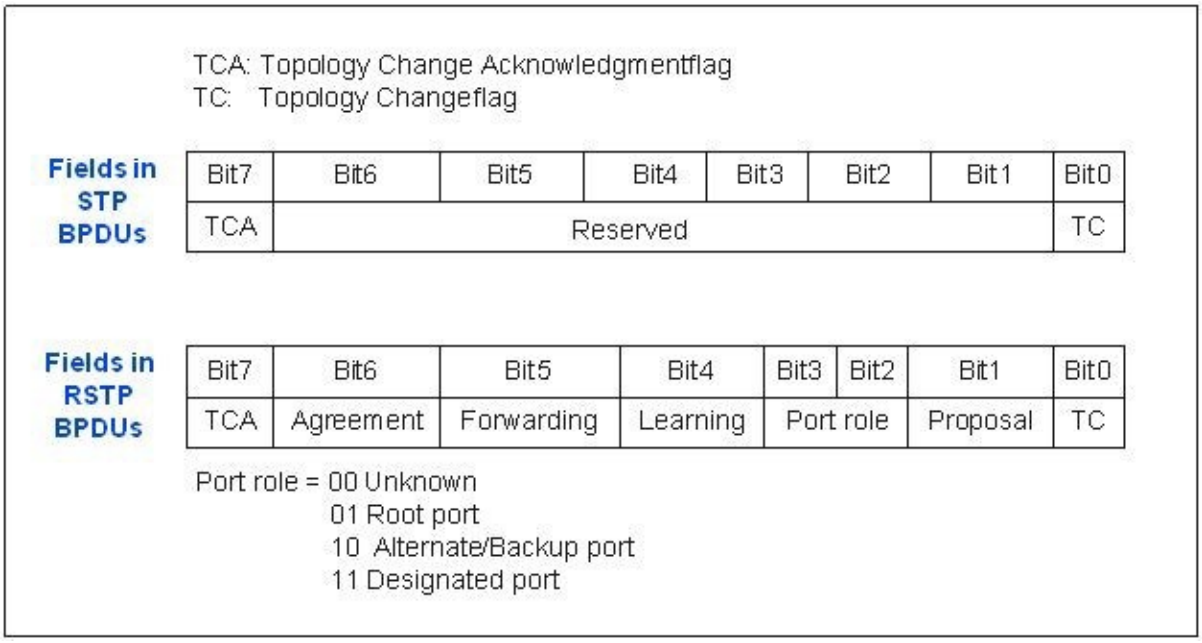


Figure 2.5: STP Convergence

Similar to STP, RSTP first selects a Root Bridge and then determines the port roles. In RSTP there are two additional port roles, alternate port and backup port. After the root and designated ports are selected the non-designated ports on non-root switches are not blocked. If the port is connected to a different switch than the root port it is turned into an alternate port, and if it's connected to the same switch it becomes a backup port. RSTP reduces the convergence time by reducing the states a port has to transition through which in STP took more than 30 seconds. RSTP also discards timers used by STP to transition to forwarding state and instead uses a sync mechanism which allows it to directly move a port to forwarding state. The combination of reduced port states, sync mechanism and shorter hello times can reduce the convergence time of RSTP to under 10 seconds.

2.4 TRILL and SPB

The development of Rapid STP (RSTP) did not completely address the needs of high-speed networks such as data centers and service provider networks, which forced the research community develop and adopt new protocols with the very complex link state routing at layer 2. Protocols like Transparent Interconnection of Lots of Links (TRILL)[11] and Shortest Path Bridging (SPB)[14] were proposed to improve the failover and convergence capabilities of STP/RSTP, but these protocols introduce complexity by adopting Intermediate system to Intermediate system (IS-IS) routing protocol into layer 2 operations . These protocols improve the performance but are not efficient to be implemented at layer 2 as introducing layer 3 routing makes these protocols complex to use. Also, the lack of granular controls and new encapsulation formats used in these protocols which are not compatible with current IP or Ethernet standard made these protocols hard to adopt.

2.5 Meshed Tree Protocol

Meshed Tree Protocol[1], which is currently under development as an IEEE standard, addresses the convergence time and performance issues of loop avoidance protocols without increasing the complexity at layer 2. Meshed Tree Algorithm (MTA)[2] allows creation and maintenance of multiple trees from one root which combine to form trees called Meshed Trees (MT). This ensures there are backup paths available in case of link/node failures or topology changes. On failure of a link (or branch) the switch can immediately fall back on another branch and frame forwarding can continue while the broken branch is pruned. MTs are implemented using a virtual identifier for each switch called MT VID which are used to maintain the path information. Multiple VIDs are stored at each switch when they received VID advertisements from their neighbors. [24]. We will discuss in detail the implementation of MTP and its components in this thesis.

2.6 Loop Avoidance Protocol with VLAN

With modern networks implementing hundreds and sometimes thousands of VLANs in their infrastructure it was essential to adapt the available loop avoidance protocols like

RSTP to work efficiently and perform better network traffic management. The standard implementation of RSTP causes major network bottlenecks around the root node as only one instance of RSTP was used for thousands of VLANs. This also had an impact on the traffic forwarding performance of the root switch as it was managing all the traffic for these VLANs. This high demand of resources around the root switch both in terms of CPU and link utilization caused the entire network performance to degrade.

To address these limitations Cisco, which is the biggest network equipment manufacturer, introduced proprietary Per-VLAN Spanning Tree Protocol (PVST) but later developed an extension in form of PVST+ which allowed it to interoperate with devices implementing IEEE standard STP protocols like RSTP [27]. PVST adopts most of its properties and functionality from RSTP but additionally allows every VLAN to run a separate instance of STP, having its own root switch and forwarding topology, allowing better and more balanced distribution of resources. Implementing PVST for networks with growing numbers of VLANs had some challenges as running separate STP instances on each VLAN can put a burden on the network and make its management complicated. To create a balance between running one STP instance for all VLAN and running separate STP instances for each VLAN, Cisco proposed the Multiple Spanning Tree Protocol (MSTP)[26][27] which was later standardized as IEEE 802.1s standard[6].

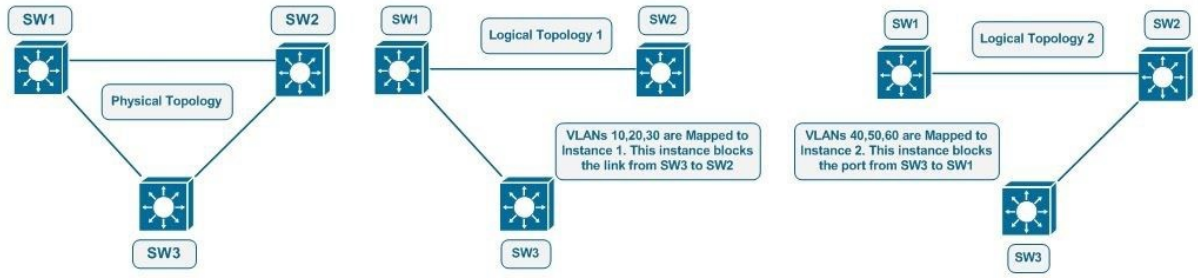


Figure 2.6: MSTP Topology

MSTP allows us to decouple STP from VLANs by running different STP instances independent of the number of VLANs. So instead of running an STP instance for every VLAN, MSTP allows us to run multiple VLAN independent STP instances and then assign each VLAN to one of the STP instances. This helps distribute the network load efficiently as we have multiple STP instances having root nodes configured on different switches while keeping the management simple as we do not have to manage hundreds or

thousands of STP instances. The core idea of MSTP is utilizing the fact that a redundant physical topology only has a small amount of different spanning-trees (logical topologies) as shown in Fig 2.6

Here as we can see, we have 2 STP instances running which divides the physical topology into two logical topologies each running its own STP instance and assigned to multiple VLANs.

3 Prior Work

Meshed Tree Protocol has been in development for the past several years and has been implemented and tested using OPNET and GENI. The current implementation of MTP on GENI uses a program developed in C language hosted and run on an internetwork of nodes which send and receive MTP Data Units (MTPDU) and use the information exchanged to form Meshed Trees. Below we will go into the detail of how this is achieved.

3.1 Meshed Tree Protocol Development

The Meshed Tree Protocol uses the Meshed Tree Algorithm (MTA) to form multiple logical trees for a physical topology. The trees are formed from a single root, which for now is manually selected. As the diagram on the left shows, there is a loop formed in the physical topology (top left). To prevent the physical loop we use MTA to form multiple logical trees from a single root as shown in Fig 3.1. The first tree is optimized for transmissions from root to A and B.

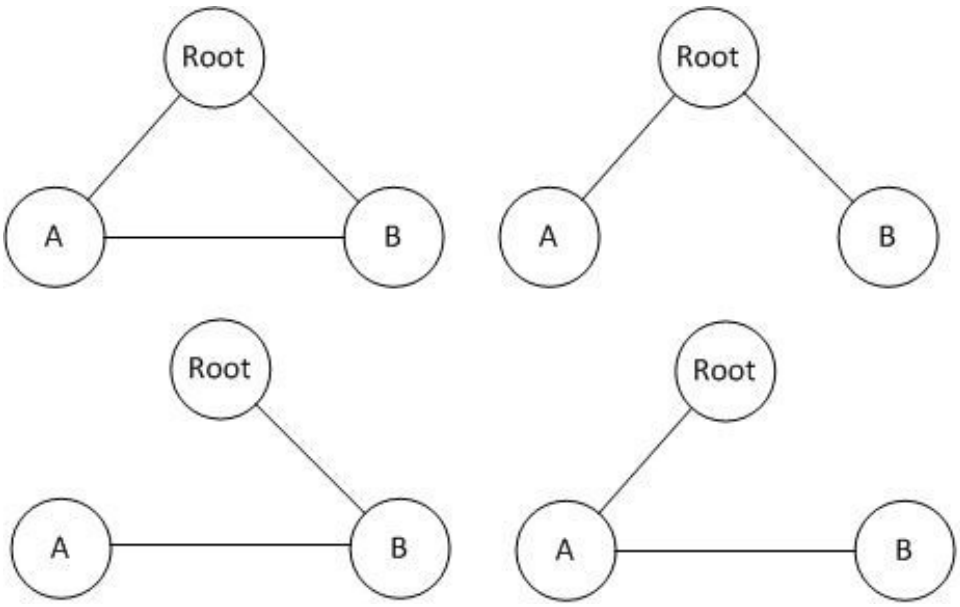


Figure 3.1: MTP Tree Formation

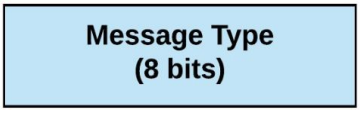
The other two trees are optimized for nodes connected to switches A and B. The tree formation process is based on assignment of multiple Meshed Tree Protocol Virtual Identifiers (VIDs) to the Meshed Tree Switch (MTS).

To generate the Meshed Trees, different MTP Data Units (MTPDU) are exchanged

between Meshed Tree Switch (MTS) to build and modify various tables containing VID information. These VIDs are ranked by best paths to root and the top three VIDs are placed in the Main VID Table. This table is used to handle broadcast traffic. Other VIDs are then placed in the Backup VID table which is used to populate the Main VID table in case a VID is removed due to a topology change or a link failure [24].

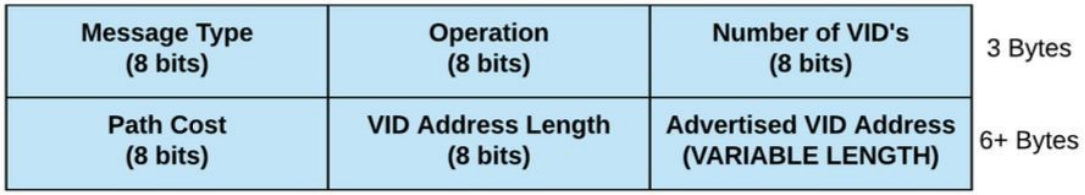
MTP Data Units

MTP JOIN



JOIN message is used by a MTS to request joining a Meshed Tree and participate in MTP. A switch which is running MTP and does not have any VID will broadcast the JOIN message on all interfaces. If a node that receives the JOIN is part of a MT it will provide a response based on which the new node will start populating its VID tables, and if the node that is not a part of a MT receives a JOIN it will be ignored. A MTP node sends JOIN messages at regular intervals till it is a part of the Meshed Tree.

MTP VID ADVERTISEMENT

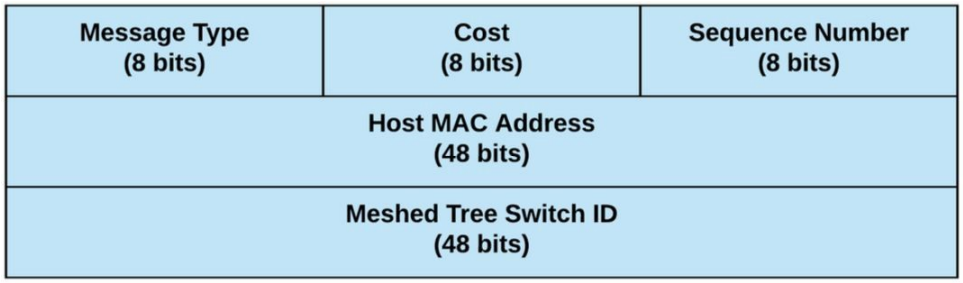


A MTS sends out ADVERTISEMENT when it receives a JOIN from another node or when there is change in topology like a link failure and the MTS updates its VID tables. This message contains VID information that is used by the receiving MTS to update its tables. This message consists of five constant and one variable length field.

- **Message Type** field provides information about the MTPDU type, for ADVERTISEMENT message it is set to 3.
- **Operation field** defines the type of action that this ADVERTISEMENT message will be used for, like addition or deletion.

- **Number of VIDs** field provides information about how many VIDs does the message include. This is used for parsing purposes as the message size can vary depending on how many VIDs it contains.
- **Path Cost** currently provides information about the hop count between the MTS and the root. This field is dynamic and can change depending on how MTP is configured and developed.
- **VID Address Length** provides length of the associated VID and is used for parsing the VID as it can have variable lengths.
- **VID Address** contains the value of the VID itself. This value is used to define the path from the MTS to the Root MTS. In the current implementation VIDs are formatted as dotted decimal notation and are calculated using the Main VIDs and the port number.

MTP HOST ADVERTISEMENT



This MTPDU is used to populate Host Tables which forward unicast messages on the most cost- effective path. HOST ADVERTISEMENT is sent when a host makes its presence known by sending a non-MTP packet to a MTS.

- **Message Type** for HOST ADVERTISEMENTS is set to a value of 4.
- **Cost** field provides a metric to measure the most efficient route to the host from the MTS, this field is dynamic and can be configured by the administrator.
- **Sequence Number** is an integer value that helps the MTS keep the most current host information by comparing the Sequence Number of the HOST ADVERTISEMENT received to its own.
- **Host Media Access Control (MAC) Address** is the IEEE 802.3 standard 48-bit hardware address of the host interface.

- **Meshed Tree Switch ID** is the MAC address with the lowest value on the MTS directly connected to the host.

MTP HELLO

Once a node has joined a MT it needs to send the HELLO message at regular intervals to the neighboring MTS' to communicate that it is active and running MTP. If no HELLO message is received by a MTS from its neighbor for a configured amount of time it is assumed that all entries received on that interface is invalid and have to be deleted.

Convergence in MTP Network

The initial convergence for a MTP network starts with a root selection. The root for the MTP network is manually selected to have better control on how the Meshed Trees are formed. Once the root is selected it waits for a MTS to send a MTPDU on one of its ports. In the Fig. 3.2 the switch on top is configured as Root and is assigned a Root ID = 1. When MTP is activated on Switch A and Switch B they realize they do not have any MTP information and start sending Join messages out from all its ports.

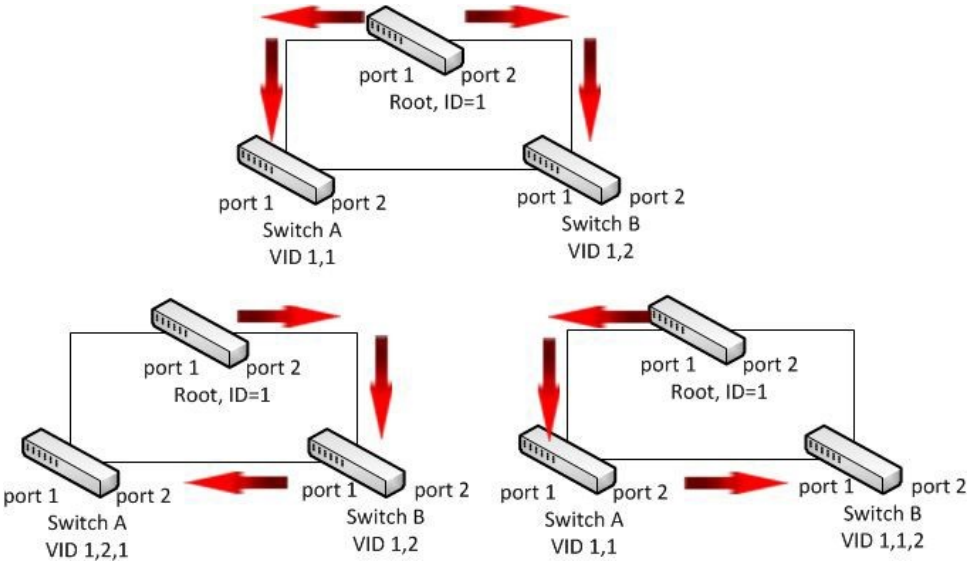


Figure 3.2: MTP Convergence

Initially Switch A and Switch B will ignore the Join messages received from each other and only the Root switch will respond to the Join message with an Advertisement message.

The VID of a switch downstream will be derived from its upstream parent VID, to which is appended the port number of the upstream or parent switch on which the child or downstream switch is connected. A switch can store multiple non-loop- forming VIDs in its VID tables which are then used to forward broadcast packets based on the path cost associated with it. In this case (Fig. 3.2) the root will assign a VID of 1.1 to Switch A and 1.2 to Switch B and will send these VIDs to respective MTS using Advertisement message. After receiving the VIDs both MTS will send back Advertisement messages with the same VID they received. When the Root receives back the Advertisement messages confirming that the VIDs were accepted, it will add this information in the Child Primary VID table (CPVID), which helps the Root maintain a list of child nodes it has. Once Switch A and Switch B receive MTP information and have been assigned VIDs, they will now start responding to Join messages from each other and populate their own VID and CPVID tables as shown in Table 3.1.

	Root	**Switch A**	**Switch B**
VID Table	1	1.1 1.2.1	1.2 1.1.2
CPVID Table	1.1 1.2	1.1.2	1.2.1

Table 3.1: MTP Tables

Each switch will have multiple VIDs assigned to it by its neighbors as explained in the Table 3.1. As we can see Switch A is assigned VID 1.1 by the root and 1.2.1 by Switch B. Similarly Switch B is assigned 1.2 and 1.1.2. If a MTS receives more than 3 VIDs it will keep the top 3 VIDs in the Main VID table and rest will be stored in Backup VID table. Having multiple VIDs allows the Meshed Tree to have all possible paths from the root to each switch and also there are multiple paths for reaching a switch from another. As the multiple paths are pre-established, failover times to redundant links are near zero.

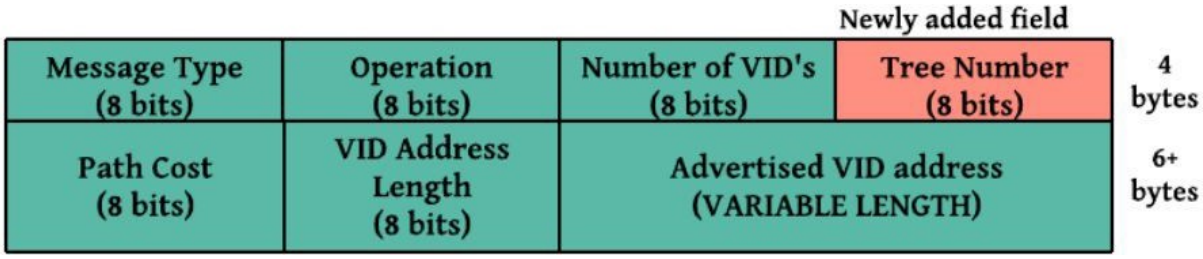
3.2 Multi Meshed Tree Protocol

While MTP is designed to efficiently manage network traffic using multiple Meshed Trees and have pre-configured backup paths in case of a failure, there is no fallback in case the MTP Root fails. Multi Mesh Tree Protocol [25] was developed as an enhancement to MTP by introducing root redundancy. The current implementation of Multi MTP allows

creation of two Meshed Trees each having its own Root. Each MTS participating in both the Meshed Trees will have two Main VID Table, two Backup VID tables and two Child PVID tables. For management of multiple trees, the MTPDUs are modified to include an additional Tree Number field. The MTS uses the Tree Number form the MTPDU to form and modify corresponding trees. Below are the updated MTPDUs for Multi MTP.

MMTP HELLO

MMTP ADVERTISEMENT



4 Methodology

4.1 Proposed VLAN Implementation in MTP

Introducing VLAN enhancement in MTP will provide multiple benefits as it will help divide the network traffic into multiple logical networks. This as discussed will add a layer of security and help scale and manage the network efficiently. For implementing VLAN in MTP we will use the current MMTP code base which allows us to have two MTP roots forming two separate Meshed Trees. The current implementation of MMTP introduces root redundancy by using the Tree Number to differentiate MTPDUs from switches which are used to form Primary and Secondary Meshed Trees. For the implementation and testing of VLAN in MTP we started by understanding the current available MMTP source code in C language and its functions and features. We added the functionality to have a user configured VLAN configuration file which will be used to generate the VLAN table. This VLAN Table will be used to filter outgoing messages based on the VLAN ID assigned to the interface. This along with Tree Number and multiple trees introduced in MMTP, we can implement VLANs in our topology and test it by creating two logical separate networks each having its own Meshed Tree. Below is the list of the functionalities incorporated into the program for VLAN implementation.

Implement and test single VLAN

- Create and store a config file to store VLAN information in predefined format.
- Create functions to read the config file.
- Modify the send functions to only send MTPDU on interfaces in that VLAN.
- Test the MTP protocol using single VLAN.

Implement multiple VLANs

- Create a dynamic nested linked list to store VID and host tables for each VLAN separately. For initial setup we will use two VLANs.

- Create functions to read, write and modify for each VLAN table.
- Assign VLAN ID based on the interface the MTPDU is received on.
- Use the VLAN ID to separate MTP traffic into two logical networks using the Tree Number. Tree 1 will only run on VLAN 1 and Tree 2 will only run in VLAN 2.
- Update MTP tables for each VLAN only using the MTPDUs received in that VLAN.

4.2 GENI Setup

For our test environment we will use the GENI which is a publicly available open-source virtual lab. We created an account on GENI and joined the MTP Project. GENI allows you to create multiple slices which act as a virtual environment to host your test setup. It gives you the flexibility to create any type of topology with multiple interconnected nodes.

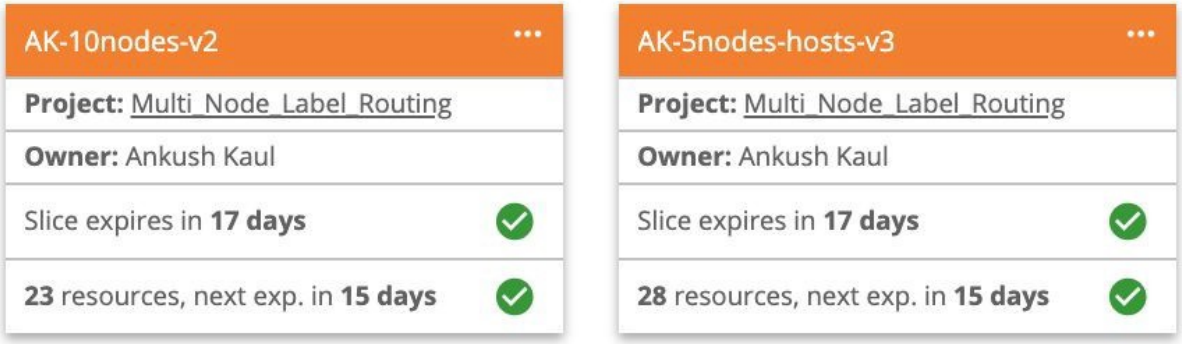


Figure 4.1: GENI Slice Information

For our setup we will create two slices as shown in Fig 4.1, one for a 5-Node and one for a 10-Node topology. Each topology consists of compute nodes connected together to emulate a network with multiple loops.

5-Node Topology

Fig. 4.2 shows our 5-Node topology, we created 5 connected nodes that will act as 5 switches labeled as SW1, SW2, SW3, SW4 and SW5 which will run MTP. We also created 8 host nodes connected to the switches.

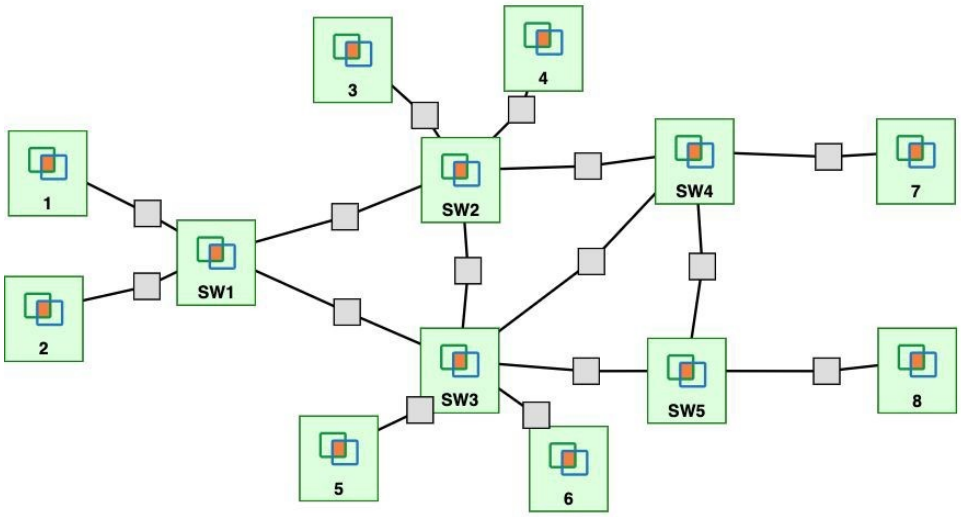


Figure 4.2: GENI 5-Node Topology with Hosts

10-Node Topology

Fig 4.2 is the 10-Node topology, here we created 10 nodes that will act as MTP switches and will run MTP code. To simplify the network and due to some GENI limitations we have not included host nodes in this topology.

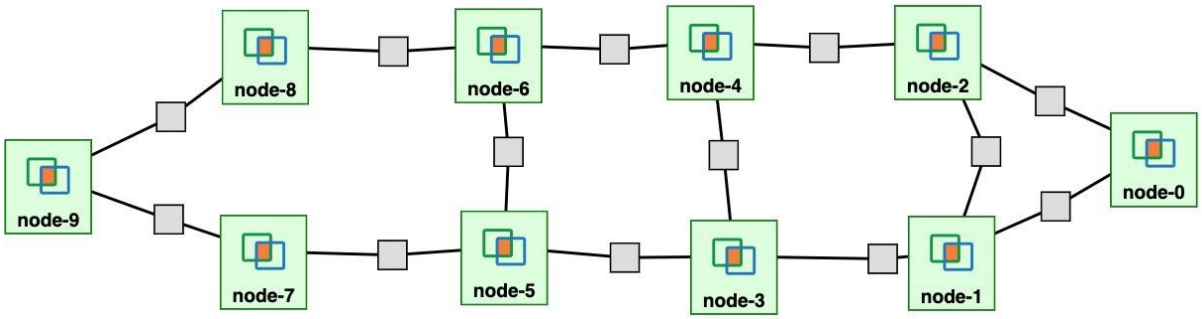


Figure 4.3: GENI 10-Node Topology

GENI allows us to securely connect to these nodes via SSH and upload and run our MTP code. It also provides details for interfaces on each node and the connectivity between them. We will need to collect all this information to use for our VLAN interface configuration and to test our output. Before we started our development we uploaded and ran the latest MMTP code on GENI to test the code and get the output for comparison. Below are the tables for the 5 Node and 10 Node topology we created.

4.3 VLAN Configuration

To implement VLAN we start with defining and creating a VLAN configuration file that the user can use to decide which interface of the MTP switch will be assigned to which VLAN. Fig 4.4 is a sample of how VLAN configuration will look like.

vlan.conf

```
1 eth1 1
2 eth2 1
3 eth3 1
4 eth4 2
5 eth5 2
```

Figure 4.4: Sample VLAN Config file

Here the interface name is represented by its logical name as per the host network configuration. We have two VLANs configured in this file, eth1, eth2 and eth3 interfaces will be part of VLAN 1 and eth4 and eth5 will be part of VLAN 2. We will create a VLAN table for each VLAN present in the configuration file, these VLAN tables will be used to get VLAN information for an interface when generating and sending MTPDUs.

4.4 Implementing Single VLAN in MTP

The first step to implement VLAN in MTP is adding the VLAN configuration file to the switches that will run MTP. For this scenario we use the topology in Fig 4.5. We added VLAN 1 to interfaces on switches SW1, SW2, SW3 and SW4 as shown in figure. The other interfaces that are not assigned any VLAN will not send any MTPDUs.

1. Read VLAN Configuration

   To initialize the VLANs we will read the configuration file and store the interface and VLAN details in a table which can be queried to check the VLAN ID for an interface while sending MTP messages

2. Initialize VLAN ID

   Before the MTP process starts we initialize the VLAN ID to a default value of 0. This will make sure that the MTP messages are not being sent till the VLAN ID is

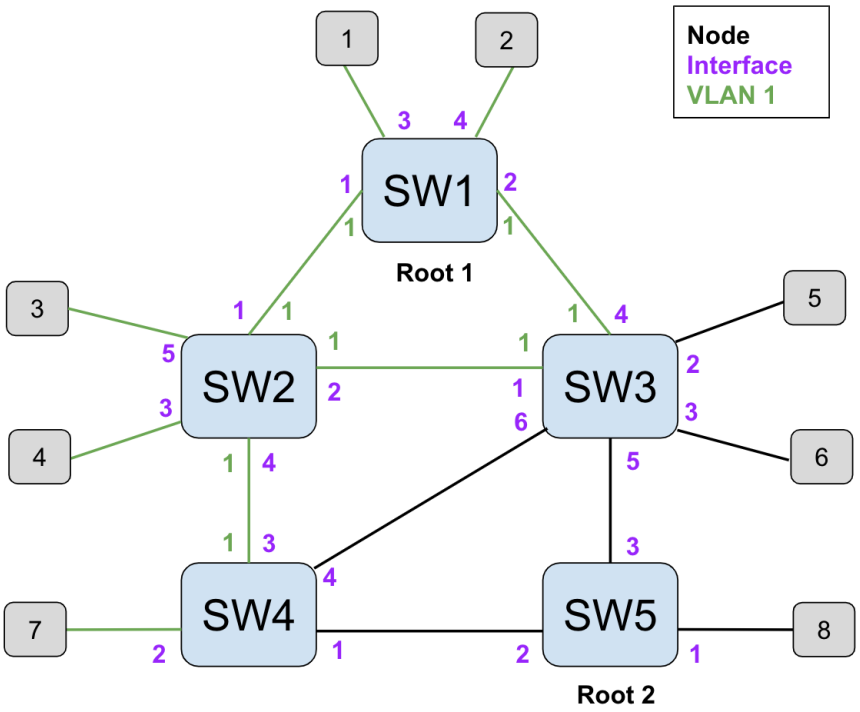


Figure 4.5: Sample 5-Node Topology with VLAN 1

set to 1 and matches the VLAN ID of the interface that the message is being sent on.

3. Get VLAN ID

   Once a MTP message is received on an interface we get the VLAN for the interface and store the VLAN in the previously initialized VLAN ID variable. This variable will be used to compare with the VLAN of the interface when sending out MTP messages.

4. Update Tables and MTP Messages

   As we have only one VLAN implemented, we will have one Main VID, Backup VID and Child VID tables. These tables will be updated when MTP Advertisements are received on VLAN 1 interfaces. After the tables are updated the switch generates and sends out MTP messages to its neighbors on VLAN 1.

4.5 Implementing Two VLANs in MTP

For implementing multiple VLANs we added logic to create and maintain multiple VLAN tables which are used to send out MTP messages. For the scope of this thesis we start with implementing two VLANs and add functionality to the available Multi MTP (MMTP) code to divide the topology into two logical networks. Switches will have one or two MTP tables depending on how many VLANs are they a part of and each VLAN will have its own Root node. Here we have two VLAN configured in our topology as shown in Fig 4.6.

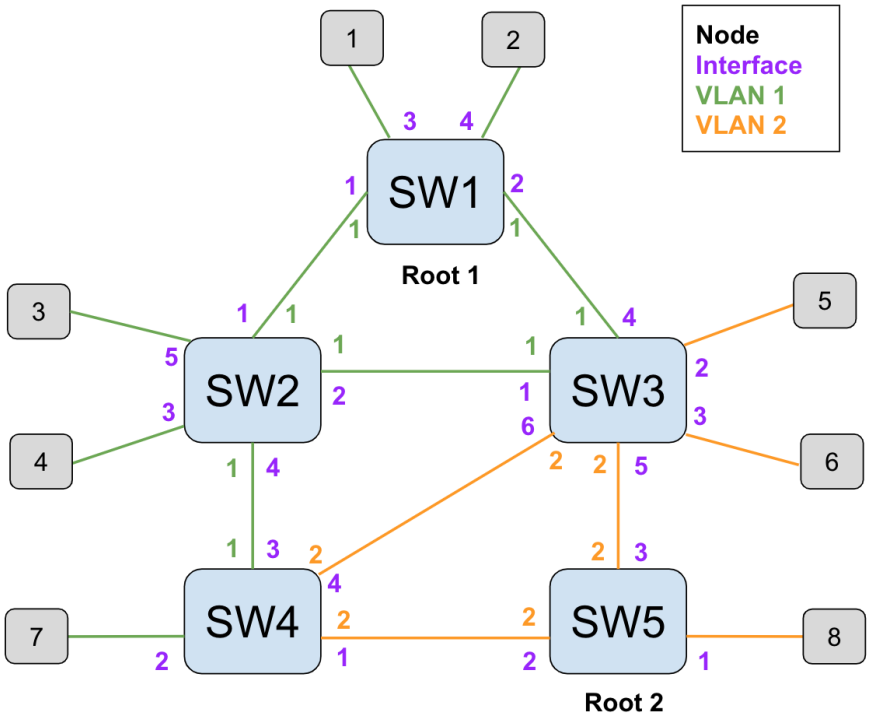


Figure 4.6: 5-Node Topology with VLAN 1 and VLAN 2

In this topology we have 5 switches with interfaces configured for VLAN 1 and VLAN 2.

SW1 and SW2

- These switches have all interfaces configured in VLAN 1 so these switches only participate in Meshed Tree within VLAN 1.
- These switches send MTPDUs for VLAN 1 only.
- They should only have one Main VID table, one Backup VID table and one Child VID table that belong to VLAN 1.

SW5

- This switch has interfaces only in VLAN 2 and participates in Meshed Tree for VLAN 2.
- This switch sends MTPDUs for VLAN 2 only.
- It should only has one Main VID table, one Backup VID table and one Child VID table that belong to VLAN 2.

SW3 and SW4

- These switches have interfaces in VLAN 1 and VLAN 2, so these switches participate in Meshed Trees for both VLAN 1 and VLAN 2.
- They send MTPDUs for VLAN 1 on interfaces in VLAN 1, and they send MTPDUs for VLAN 2 on interfaces in VLAN 2.
- Both the switches should have two Main VID table, two Backup VID table and two Child VID table, one for each VLAN.

For implementing two VLANs we expand and introduce additional functionality to our code that we used for Single VLAN implementation.

1. Read VLAN Configuration

Figure 4.7: Sample VLAN Config file

For the two VLAN implementation we implemented a function to create multiple VLAN lists depending on the number of VLANs in the configuration file. For example the VLAN configuration file in Fig 4.7 has two VLANs, so the function will create two different VLAN lists and add respective interfaces to the VLAN list. Interfaces eth1, eth2 and eth3 will be added to VLAN 1 list; eth4 and eth5 will be

added to VLAN 2 list.

2. Initialize VLAN on Root 1 and Root 2

   When the MTP is run on Root 1 and Root 2, they will start the formation of two independent Meshed Trees for VLAN 1 and VLAN 2. As the Root and the VLAN have been hardcoded, the VLAN ID is set to match the Root ID, VLAN ID for Root 1 node is set to 1 and for Root 2 is set to 2. After the VLAN ID is initialized the VLAN configuration is read and the VLAN 1 and VLAN 2 tables are populated on both the Root nodes, with the interfaces assigned to VLAN 1 and VLAN 2. Each node will then send two types of MTPDUs.

   VID update messages is sent by each node on interfaces in its VLAN, so Root 1 will send VID updates on VLAN 1 interfaces and Root 2 will send VID updates on VLAN 2 interfaces.

   Join messages are sent if there are interfaces on the Root node that are not part of the Root node VLAN. For example if Root 1 had an interface in VLAN 2 it would send a Join message on that interface. That is not the case in our scenario so no Join messages will be sent by the Root nodes.

3. Initialize VLAN on Non-Root nodes

   For the non-Root nodes, the VLAN ID is kept at its default value of 0. When these nodes are initialized, they will send out Join messages on all their interfaces to begin with. In this scenario SW2, SW3 and SW4 will be sending out Join messages on their interfaces.

4. Start MTP

   Once all the nodes, Root and non-Root nodes in both the VLANs, are initialized the MTP function will start to run on all nodes. First the link failure logic will check for link failures and send out update messages on all interfaces. Next the periodic timer checks and sends out periodic messages, which is used to keep a link active, and join messages in case the node is not participating in MTP yet. After this the receive buffer on the node is checked and read for any received MTP packets. Based on the type of packet received suitable action are taken as described below.

5. Receive MTPDU

   Once a MTP message is received on an interface we get the VLAN for the interface and store the VLAN in the previously initialized VLAN ID variable. This variable will be used to compare with the VLAN of the interface when sending out MTP messages on configured interfaces. If the message is received on a VLAN 1 interface then the VLAN ID will be set to 1 and if the message is received on VLAN 2 interface the VLAN ID is set to 2. The Message Type field is checked on the received MTPDUs and one of the below actions is performed.

   Join Message

   After a Join message is received, the switch checks its VID table for VID entries, if VID entries are present then the switch generates a VID Advertisement message which is sent out of interfaces with VLAN matching the previously set VLAN ID, VLAN of the interface the MTPDU was received on.

   Hello Message

   If a Hello Message is received we get the Tree Number from the message and the switch generates and sends out its own Hello Messages out from all ports of the VLAN that the meshed tree belongs to. For example, in the current scenario if the Periodic Message was received on an interface which is in VLAN 1 and part of Meshed Tree 1, the switch sends out its own Periodic Messages on all interfaces in VLAN 1. The switch also updates its HELLO timers after receiving a Periodic Message, this acts like a keep alive mechanism.

   VID Advertisement

   Each MTP switch receives and processes VID Advertisement messages which are used to generate and update the VID tables on the switch. VID Advertisements are sent in response to a Join Message and are critical in the formation of the Meshed Trees. In our VLAN implementation when a VID Advertisement is received we check the Tree information which we get from the Tree Number field and the VLAN information from the receiving interface. After the Tree Number and VLAN information is gathered the switch adds or deletes the VIDs provided in the message based on the operation specified in the Operation field. These actions are performed

for both VLAN 1 and VLAN 2 trees and respective VID tables are updated. After the switch tables are updated the switch now generates and sends out its own VID Advertisement to interfaces in VLAN 1 or VLAN 2.

6. MTP Messages and VID Tables

   The combination of above described MTP Messages which are used to communicate between MTS form the backbone of the Meshed Trees and allow creation, update and deletion of VIDs and VID tables which are then used to forward the network traffic. By implementing the VLANs we are able to restrict and isolate the network traffic to interfaces within those VLANs and create and maintain Meshed Trees per VLAN.

Fig 4.7 and 4.8 explains this implementation of VLANs for MTP using a flowchart. This flowchart shows the step by step flow of how the MTP process starts, when the MTPDUs and received and the actions taken for each type of MTP Message.

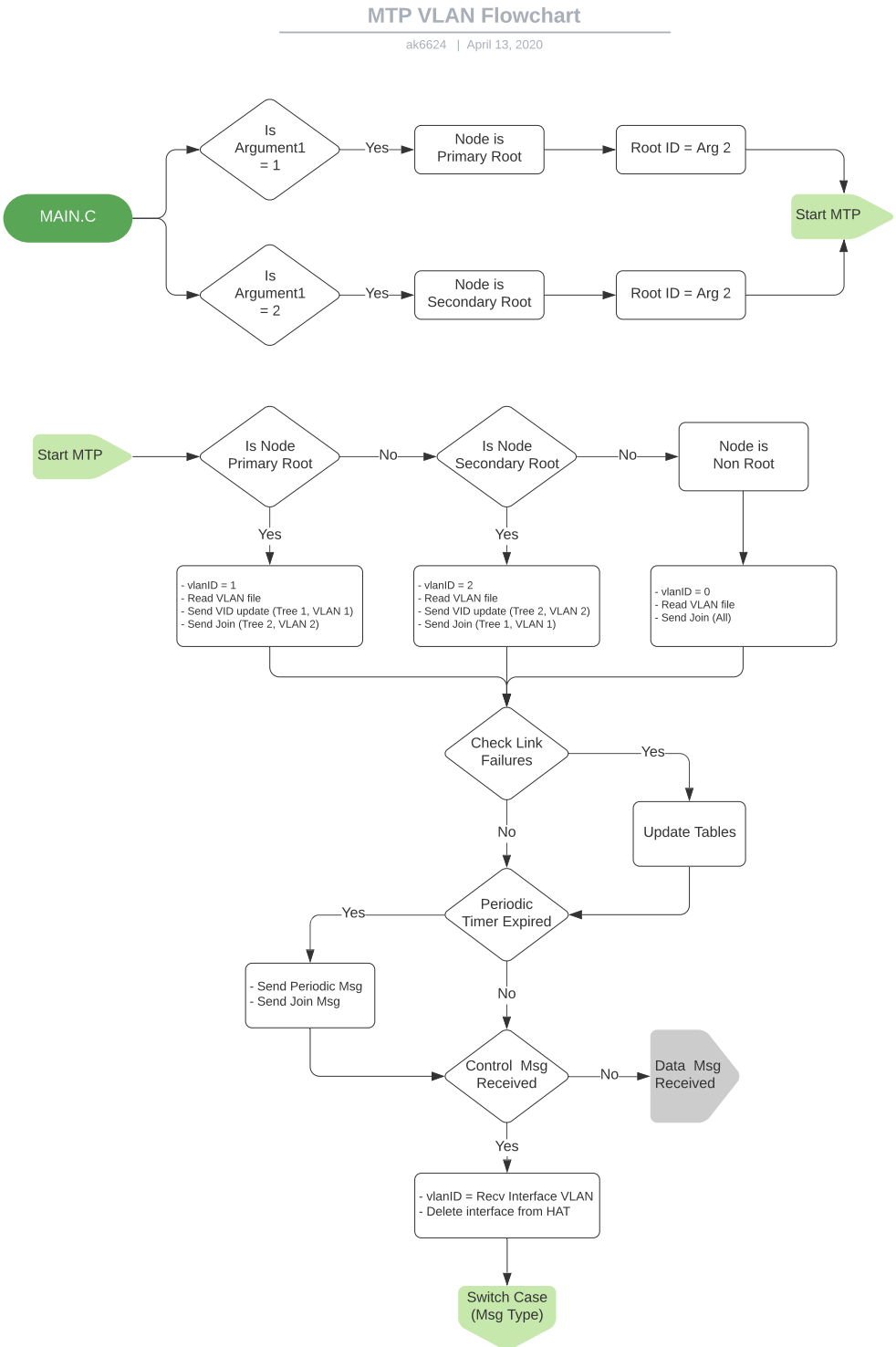


Figure 4.8: MTP VLAN Flowchart - Part 1

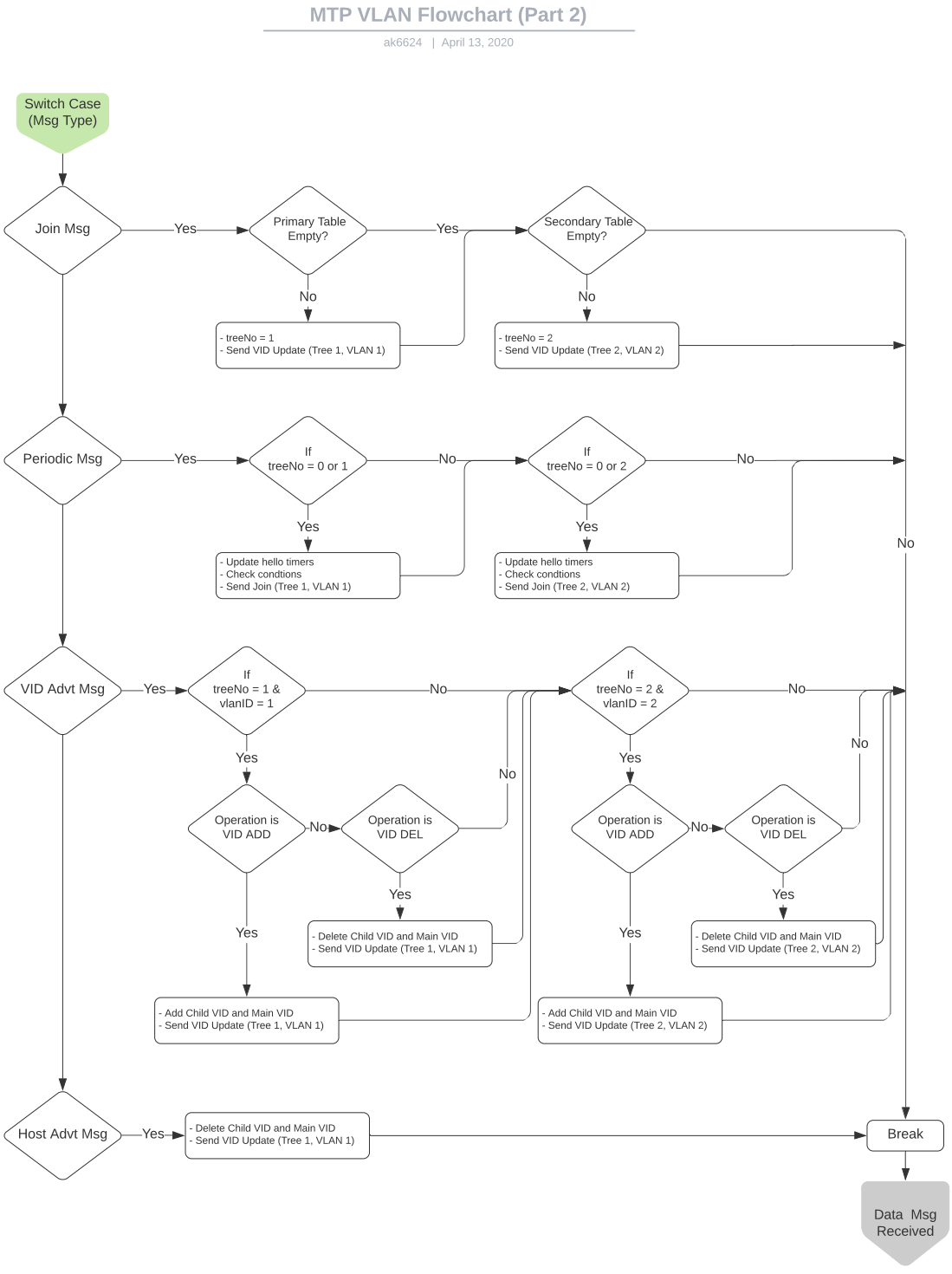


Figure 4.9: MTP VLAN Flowchart - Part 2

5 Results

This section will demonstrate and analyze the results for different topologies implemented in GENI as discussed in previous sections. The below results are collected from the console output after running the MTP code on all the nodes in GENI. To generate these results we implemented additional functionality to output VLAN tables and VID Tables for individual VLANs. For each of the below topologies we have shown the output in form a table where we listed the VIDs for each node belonging to the VLANs we have configured. We also analyze and discuss the output results for each topology. Each switch will have VLAN 1 and/or VLAN 2 tables depending on the VLANs configured on that switch. A switch with only a single VLAN will participate in only one Meshed Tree formation and have only one set of VID tables, while a switch with interfaces configured for multiple VLANs will participate in two Meshed Tree formations and will have separate VID tables for both Meshed Trees.

The output here will show us that we were able to successfully implement VLANs in MTP and with the help of separate VID tables, for each Meshed Trees, we can segregate the network traffic.

5.1 5 Node Topology - One VLAN

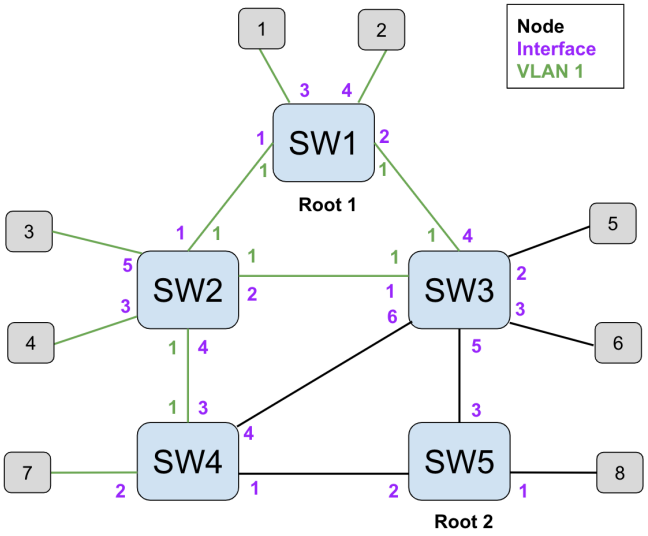


Figure 5.1: 5-Node Topology with VLAN 1

For the first scenario with one VLAN we added VLAN config files to switches SW1, SW2,

SW3 and SW4. This config file will assign VLAN 1 to certain ports as shown in Fig. 5.1. Any switch or port not assigned a VLAN will not participate in the MTP process as we restrict the MTP messages only to VLAN 1. This is the first step for implanting VLAN as we try to experiment and understand the best way to filter MTP messages.

VID Table Output

SW1	**SW2**	**SW3**	**SW4**	**SW5**
1	1.1 1.2.1	1.2 1.1.2	1.1.4 1.2.1.4	

Table 5.1: VID tables output for VLAN 1

Observation

We successfully restricted the Meshed Tree formation to the ports configured in VLAN by filtering the MTP messages at the switch ports using the Root ID and the VLAN. Below are few observations made.

- As no VLAN is configured for Switch 5, it will not have any VID tables.
- All interfaces of Switch 1 and Switch 2 are in VLAN 1.
- Switch 3 and Switch 4 only have some interfaces in VLAN 1 and other interfaces do not have any VLAN assigned so it will not send or receive MTP Messages.

5.2 5 Node Topology - Two VLANs

For implementing multiple VLANs we introduced the VLAN config on all the switched and limited our VLANs to two. Here all the switches and ports will participate in the Meshed Tree formation, Fig 5.2.

VID Table Output

VLAN	**SW1**	**SW2**	**SW3**	**SW4**	**SW5**
1	1	1.1 1.2.1	1.2 1.1.2	1.1.4 1.2.1.4	
2			2.3 2.2.4	2.2 2.3.6	2

Table 5.2: VID tables output for VLAN 1 and VLAN 2

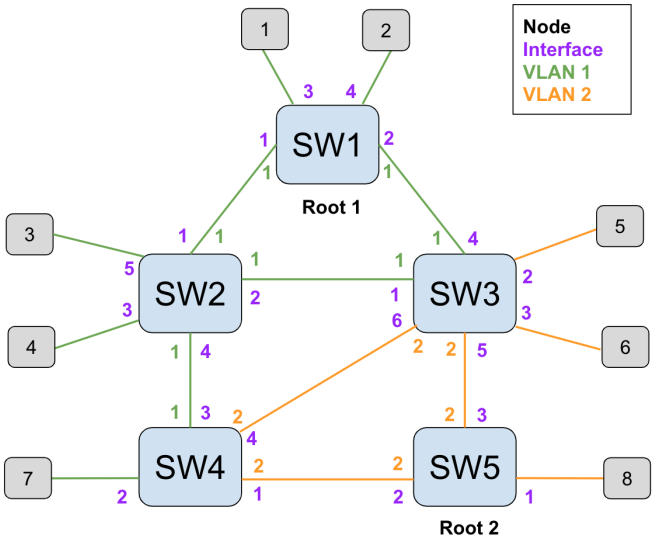


Figure 5.2: 5-Node Topology with VLAN 1 and VLAN 2

Observation

Output in Table 5.2 shows we successfully implemented two VLAN which allowed creation of two Meshed Tress in our network topology. This allows us to restrict network traffic for better network management. Below are observations made here.

- Switch 1 and Switch 2 have only VLAN 1 configured, these switches will only have VLAN 1 VID tables.
- Switch 3 and Switch 4 have interfaces in both VLANs so these switches will have VLAN 1 and VLAN 2 VID tables
- Switch 5 will only have VLAN 2 VID tables as it only have VLAN 2 configured.

5.3 10 Node Topology - Two VLANs

We also tested the two VLAN configuration on a bigger network with multiple loops. Here for simplicity we did not include the host nodes, Fig. 5.3.

VID Table Output

Observation

Similar to the previous output we used VLAN config files on each switch to configure the ports and implement VLAN within MTP to create two different Meshed Tress limiting

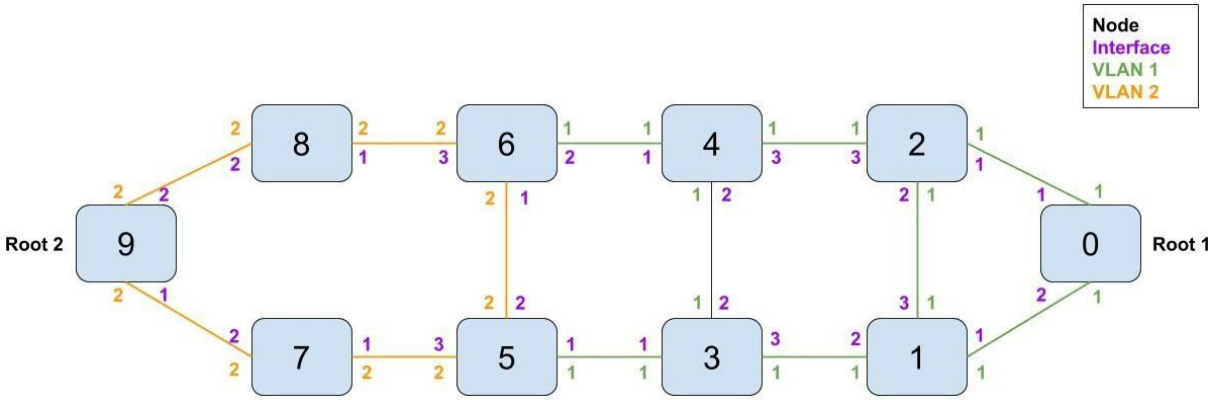


Figure 5.3: 10-Node Topology with VLAN 1 and VLAN 2

Node 9	**Node 8**	**Node 7**	**Node 6**	**Node 5**	**Node 4**	**Node 3**	**Node 2**	**Node 1**	**Node 0**
			1.1.3.1 1.2.3.3.1 1.2.2.2.1	1.2.2.1 1.1.2.2.1 1.1.3.2.1	1.1.3 1.2.3.3 1.2.2.2	1.2.2 1.1.2.2 1.1.3.2	1.1 1.2.3 1.2.3.3	1.2 1.1.2 1.1.2.2	1
2	2.2 2.1.1.2.3	2.1 2.2.1.1.3	2.2.1 2.1.1.2 2.1.1.2.3	2.1.1 2.2.1.1 2.2.1.1.3					

Table 5.3: VID tables output for VLAN 1 and VLAN 2

and isolating the network traffic. Observations made here are as follows.

- Nodes 0 to 4 have only VLAN 1 configured so for these nodes only the VLAN 1 VID table is generated.
- Nodes 7 to 9 have only VLAN 2 configured so for these nodes only the VLAN 2 VID table is generated.
- For nodes 5 and 6 which participate in both VLAN 1 and 2 VID tables are generated for both VLANs.

6 Conclusion

MTP provides a reliable and efficient alternative to current loop avoidance protocols. As per the findings in recent papers show MTP providing better performance than RSTP which is a big step forward. Further work on Multi Meshed Tree makes MTP highly reliable without sacrificing the performance. This thesis tries to further strengthen the Meshed Tree Protocol by implementing VLANs which can be used for better security and network management. Following are the list of features implemented:

- Read and store VLAN config for MTP ports.
- Create nested link list for VLANs in the config file.
- Separate the Primary and Secondary Root into separate VLANs.
- Restrict the MTP messages sent based on interface VLAN.
- Populate the VID tables to include only nodes in a VLAN
- For nodes in multiple VLANs populate multiple VID tables

6.1 Future Work

This thesis lays the groundwork for understanding and developing VLAN implementation in MTP. Using nested linked lists implemented in VLAN code we can further expand VLANs to dynamically create multiple MTP Tables for n number of VLANs in the VLAN config file. Another addition as the next steps of this thesis would be to introduce multiple roots within each VLAN for added redundancy.

References

[1] K. Sharma, B. Hartpence, B. Stackpole, D. Johnson, and N. Shenoy, "Meshed tree protocol for faster convergence in switched networks," Apr. 2014, pp. 90–95.

[2] K. Sharma, "Implementation of the meshed tree algorithm on a switched network," Master's thesis, Rochester Institute of Technology, 2016.

[3] IEEE Computer Society, IEEE standard for local and metropolitan area networks: media access control (MAC) bridges, IEEE Std. 802.1D. New York, N.Y.: Institute of Electrical and Electronics Engineers, 2004.

[4] "Overview of VLANs." Cisco Systems, [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/20ew/configuration/guide/config/vlans.pdf.

[5] IEEE Computer Society, Local and Metropolitan Area Networks - Common Specifications - Part 3: Media Access Control (MAC) Bridges: Amendment 2 - Rapid Reconfiguration IEEE Std 802.1w. Institute of Electrical and Electronics Engineers, 2001.

[6] IEEE Computer Society, Amendment to 802.1Q Virtual Bridged Local Area Networks: Multiple Spanning Trees. IEEE Std 802.1s (Amendment to IEEE Std 802.1Q, 1998 Edition). Institute of Electrical and Electronics Engineers, 2002.

[7] "Understanding Rapid Spanning Tree Protocol (802.1w)." Cisco Systems, Aug. 2017, [Online]. Available: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html.

[8] R. Perlman, "An algorithm for distributed computation of a spanningtree in an extended LAN," SIGCOMM Comput. Commun. Rev., vol. 15, no. 4, pp. 44–53, Sep. 1985.

[9] "Understanding Spanning-Tree Protocol Topology Changes." Cisco Systems, Jun. 2016, [Online]. Available: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/12013-17.html.

[10] "Understanding Rapid Spanning Tree Protocol (802.1w)." Cisco Systems, Aug.

2017, [Online]. Available: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html.

[11] R. Perlman and J. Touch, "Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement." RFC 5556, May 2009, [Online]. Available: https://tools.ietf.org/html/rfc5556.

[12] D. Eastlake, Huawei, R. Perlman, Intel Labs, and Cisco, "Transparent Interconnection of Lots of Links (TRILL)." RFC 7177, May 2014, [Online]. Available: https://tools.ietf.org/html/rfc7177.

[13] R. Perlman, Intel Labs, D. Eastlake, and Cisco, "Routing Bridges (RBridges): Base Protocol Specification." RFC 6325, Jul. 2011.

[14] IEEE Computer Society, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks–Amendment 20: Shortest Path Bridging IEEE Std. 802.1aq. Institute of Electrical and Electronics Engineers, 2012.

[15] IEEE Computer Society, IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks. IEEE Std. Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011). Institute of Electrical and Electronics Engineers, 2014.

[16] K. Sharma, B. Hartpence, B. Stackpole, D. Johnson, and N. Shenoy, "Meshed Tree Protocol for Faster Convergence in Switched Networks," 2014, pp. 20–24.

[17] P. Lapukhov, "Understanding STP and RSTP Convergence." INE, 2010, [Online]. Available: https://cdn2.hubspot.net/hubfs/3985396/Imported-Blog-Media/understanding-stp-rstp-convergence-3.pdf.

[18] Linux Foundation, "Open vSwitch," 2019. https://www.openvswitch.org.

[19] Geni, "Geni: Exploring networks of the future." https://www.geni.net/.

[20] The Scapy Community, "Scapy." https://scapy.net/.

[21] P. Willis and N. Shenoy, "The Evaluation of a Meshed Tree Protocol on the GENI Testbed. ," 2018, pp. 1–2.

[22] P. Willis and N. Shenoy, "A Meshed Tree Protocol for loop avoidance in switched networks," 2019, pp. 303–307.

[23] C. Zhang and S. Ji, A SNMP-base broadcast storm identification method in VLAN. Atlantis Press, 2013.

[24] P. Willis, "A Performance Analysis of the Meshed Tree Protocol and the Rapid Spanning Tree Protocol.," Master's Thesis, Rochester Institute of Technology, 2019.

[25] S. Rudroju, "Root Failure Analysis In Meshed Tree Networks," Master's Thesis, Rochester Institute of Technology, 2020.

[26] "Understanding Multiple Spanning Tree Protocol (802.1s) ." Cisco Systems, Apr. 2007.

[27] P. Lapukhov, "Unserstanding MSTP," 2010. https://blog.ine.com/2010/02/22/understanding-mstp.